

# Trends in Server Platform Security

## Platform Security Summit 2019

**Rob Wood**



# Who am I

Currently:

NCC Group - VP of Hardware and Embedded Security Services

Previously:

Motorola Mobility - Security Architect

BlackBerry - Security Research Group

Research in Motion - OS/FW Security Dev

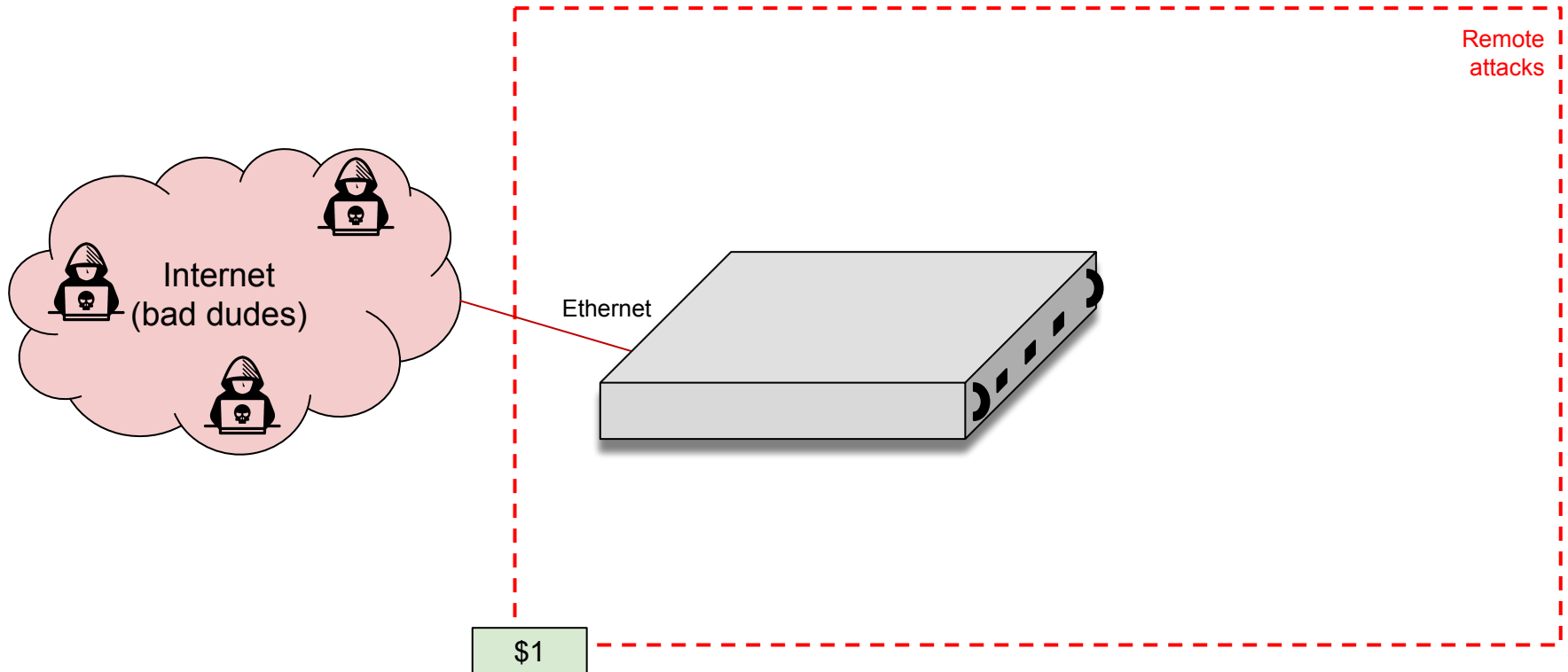
Always:

Fights for the user



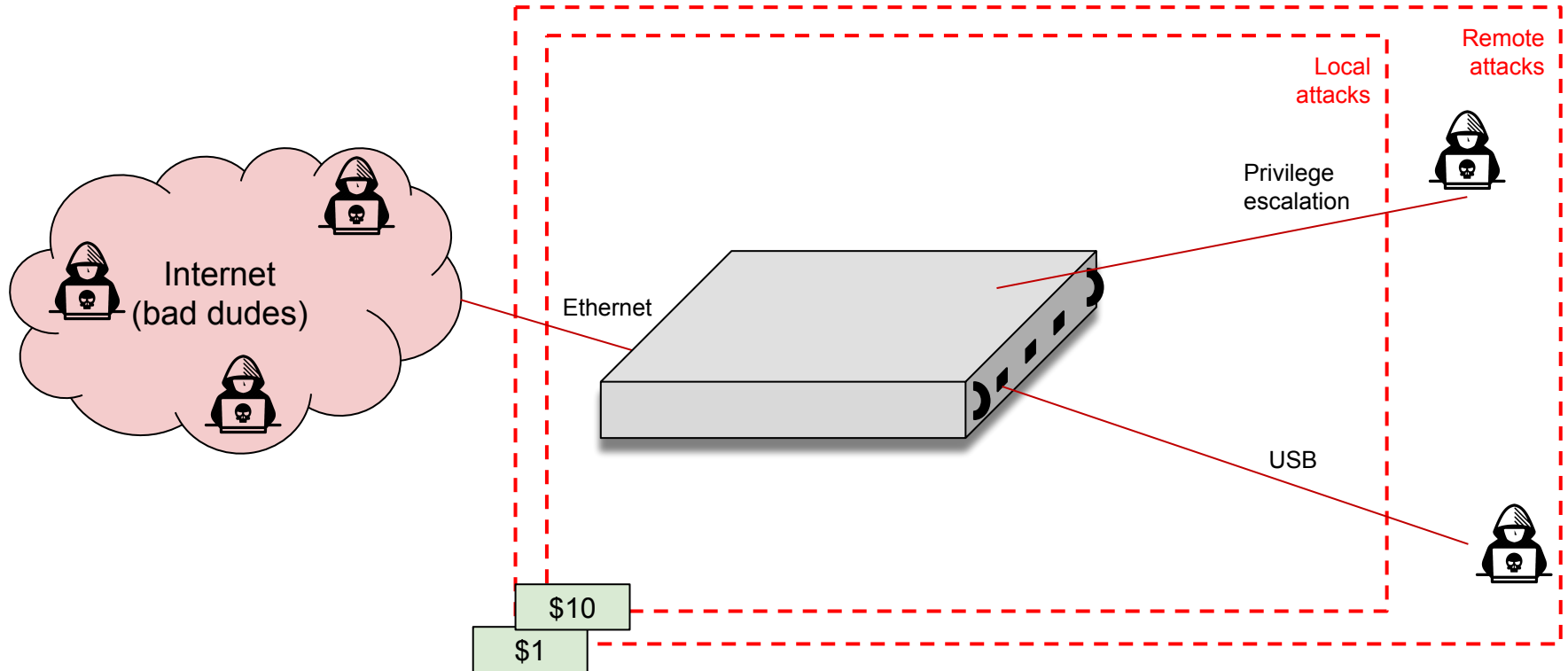
# Platform Security

Threat model: where is your attack surface?



# Platform Security

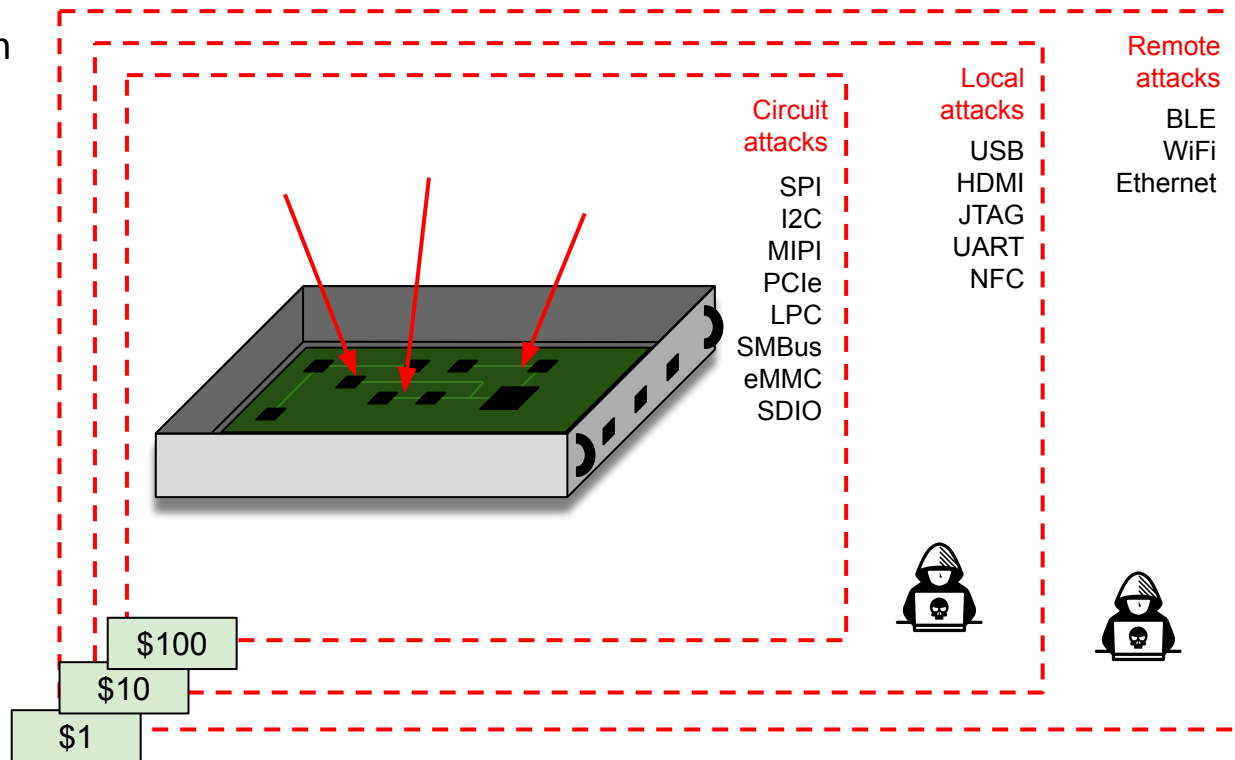
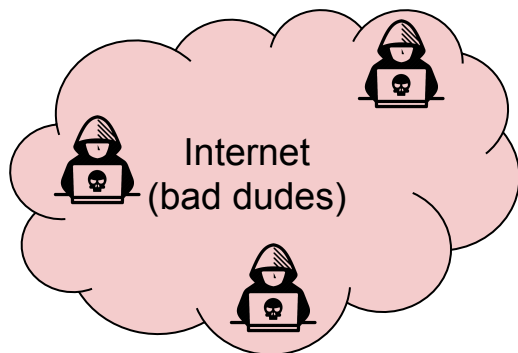
Threat model: where is your attack surface?



# Platform Security

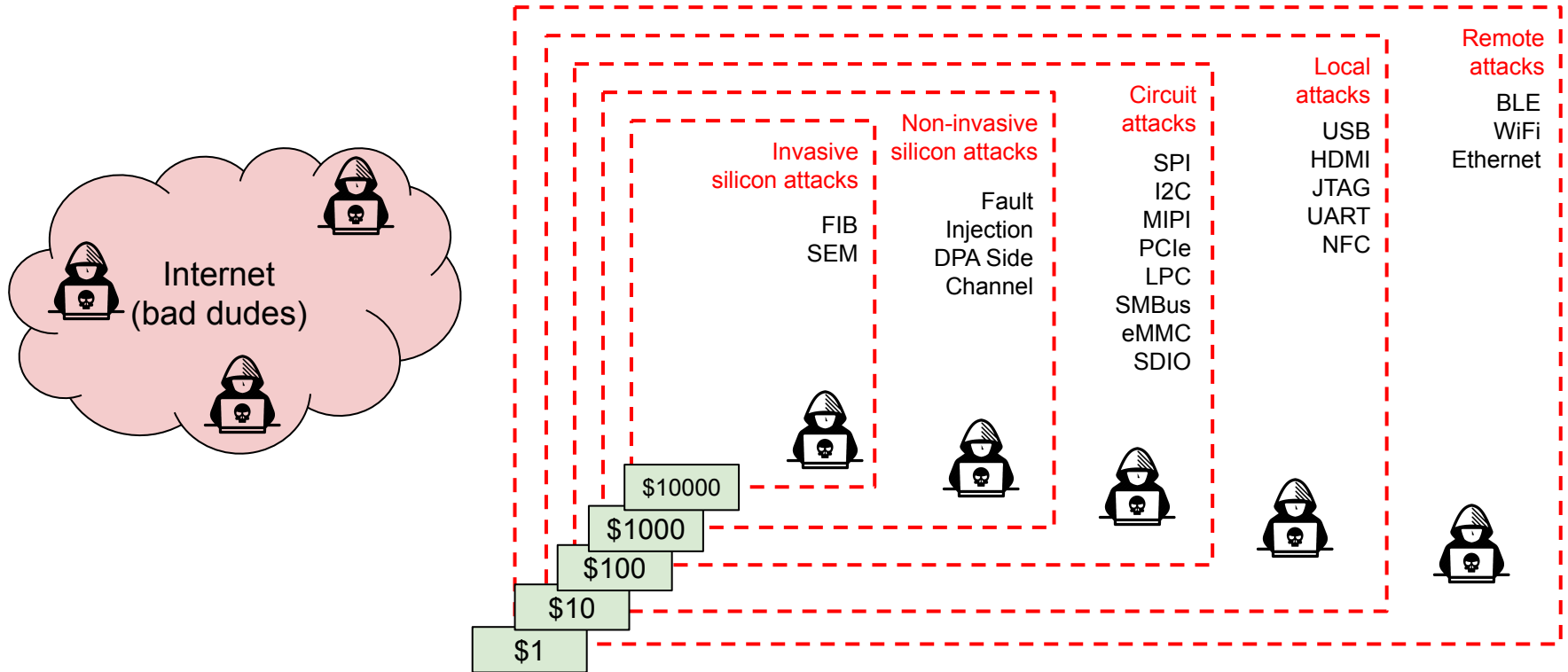
Threat model: where is your attack surface?

It gets deeper depending on the threat actor!



# Platform Security

Threat model: where is your attack surface?



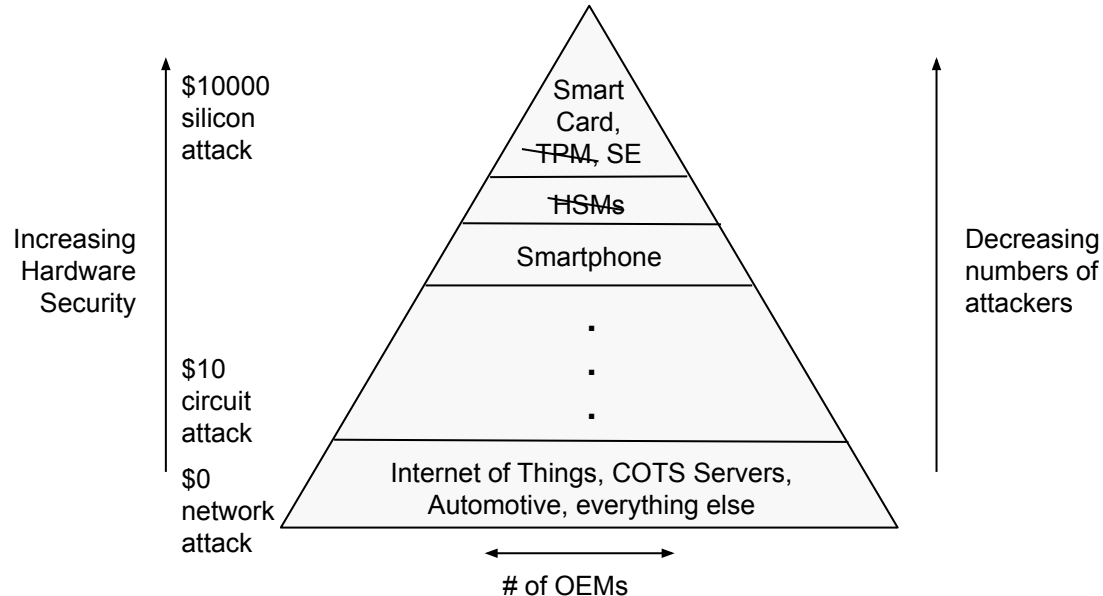
# Why do we care?

- Third party data centres, with unknown security controls
  - Edge network and CDN deployments
  - Cloud providers
    - Need to avoid/prevent malware persistence
    - Hypervisor escape may give access to firmware
    - Bare-metal cloud gives access to firmware
- Supply chain attacks
  - “bloomberg/supermicro” [\\*](#)



# Hardware Security Ecosystem

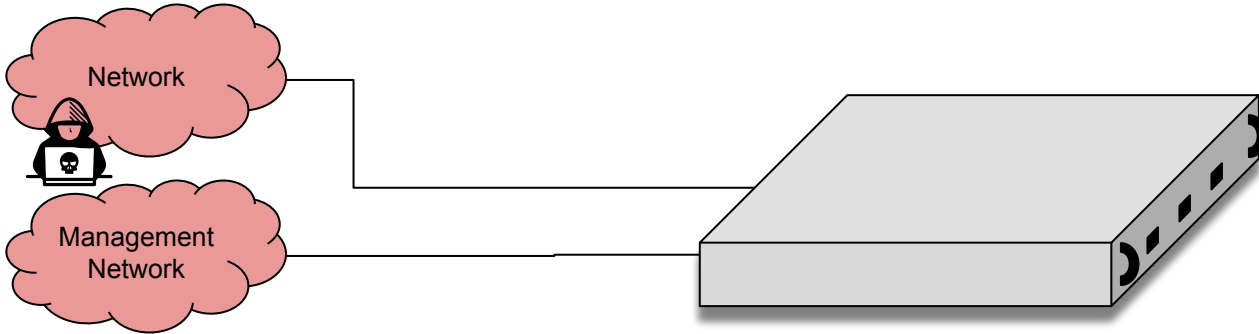
- Privilege escalation in SW through HW modification and abuse
- Data extraction
- cost: higher barrier-to-entry for attackers due to equipment costs
- maturity: HW security ~15 years behind the state of SW security, with very few exceptions



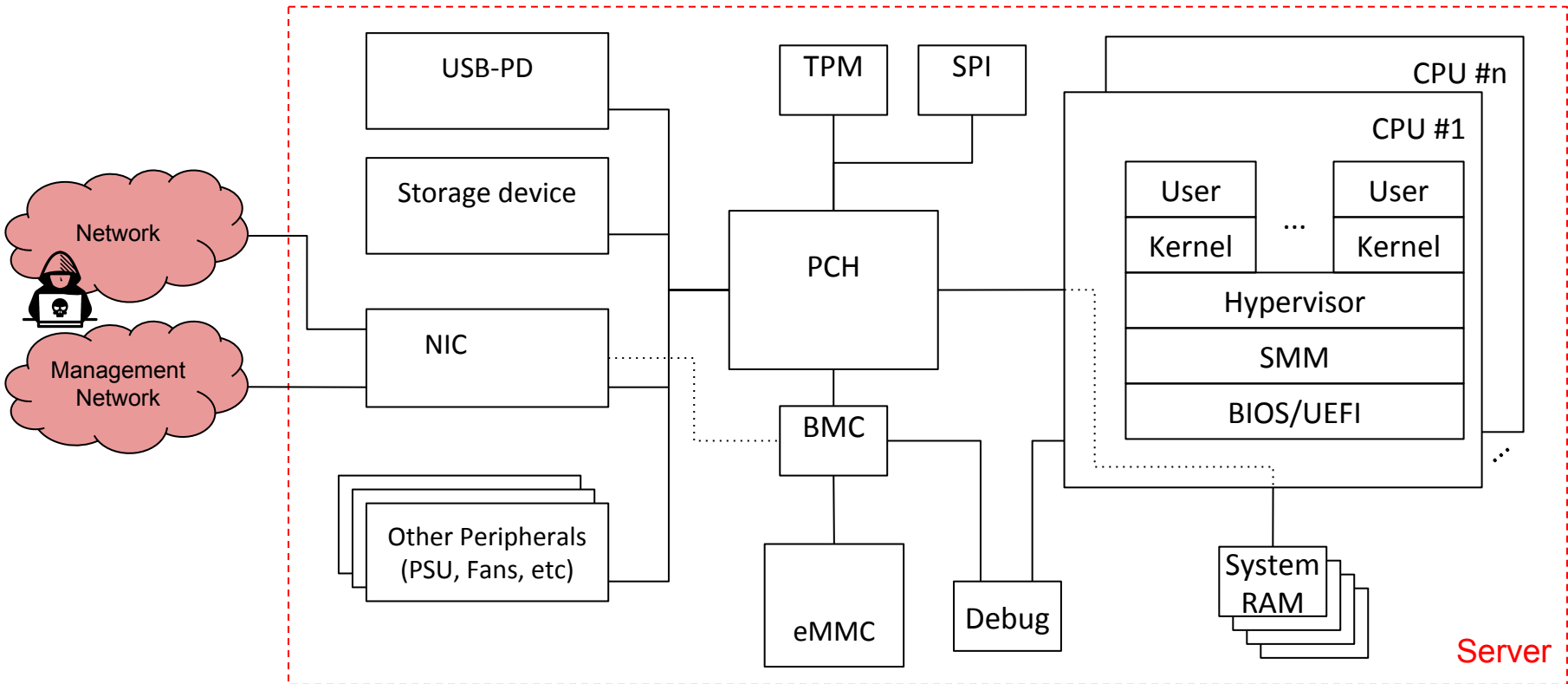


# Server Architecture

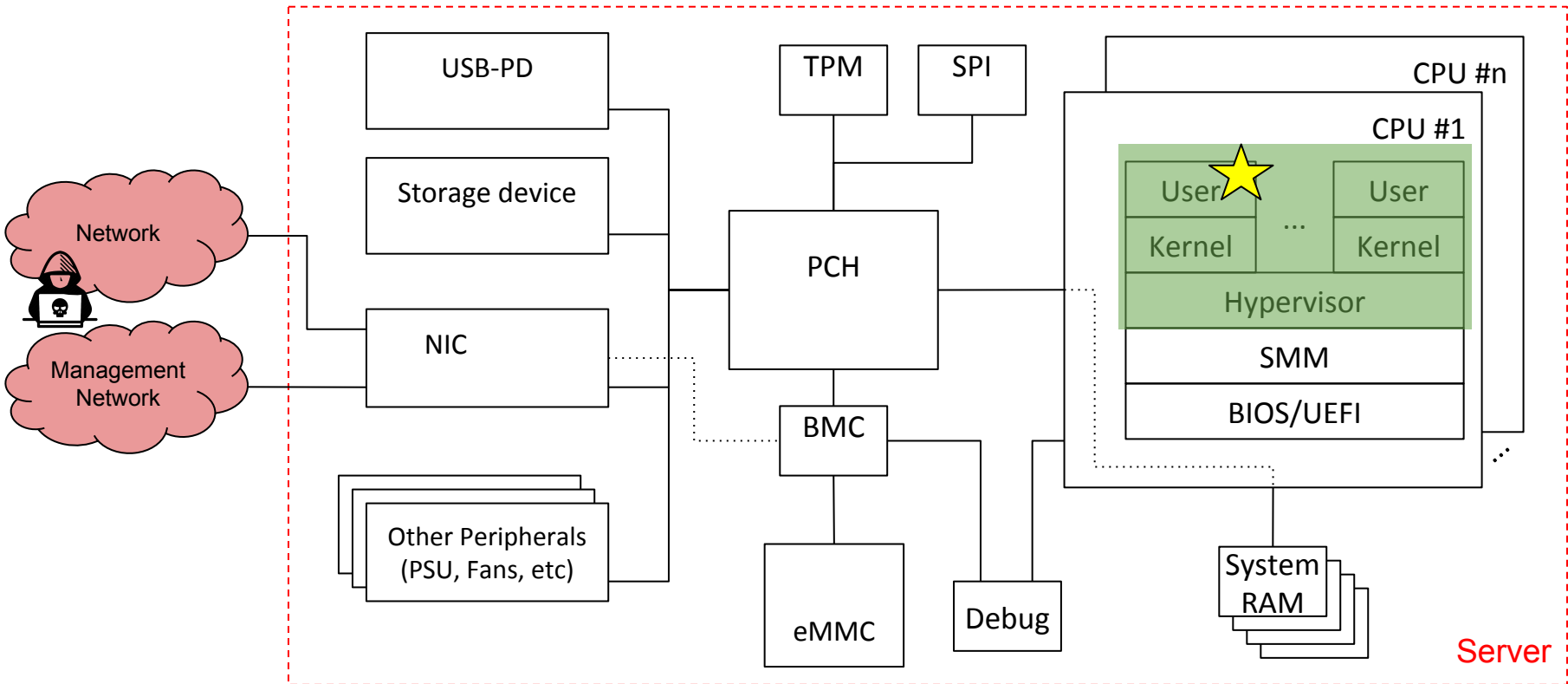
---



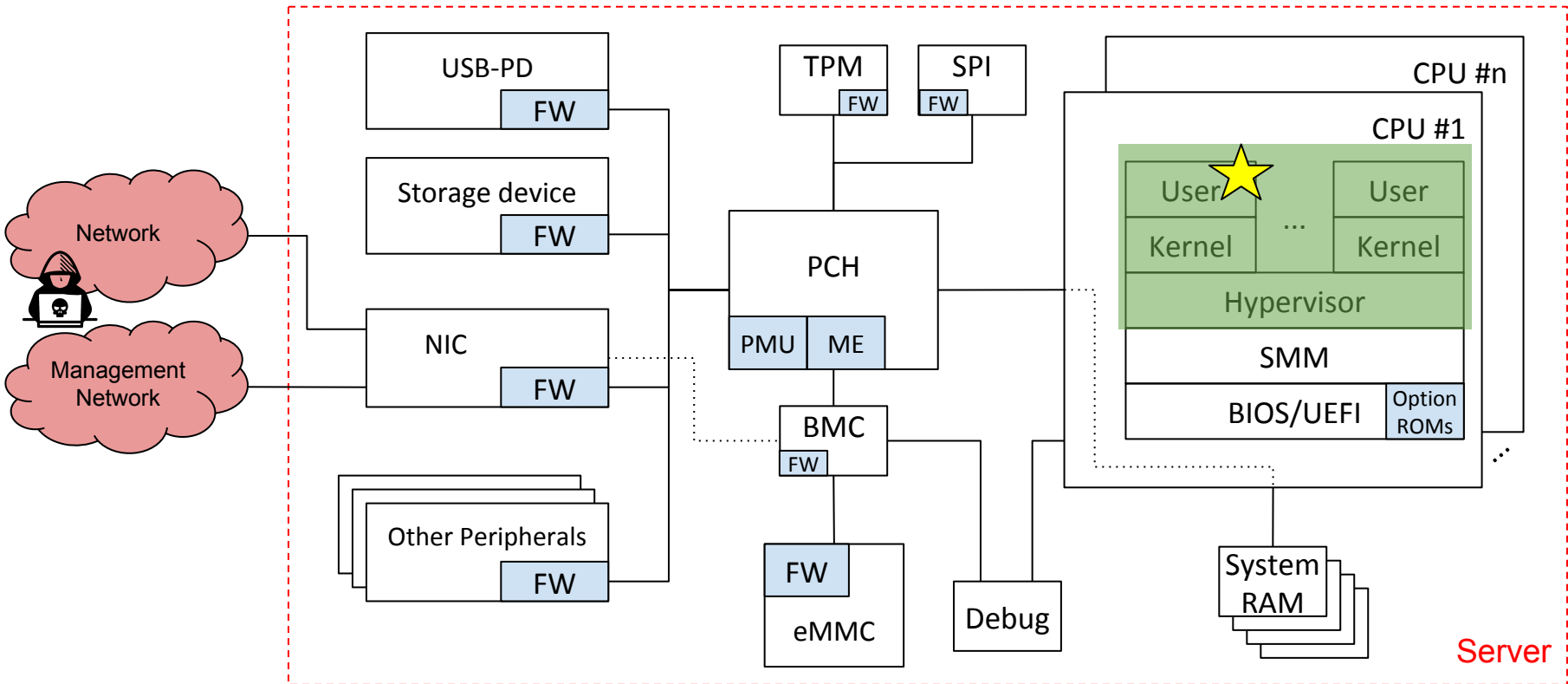
# Server Architecture



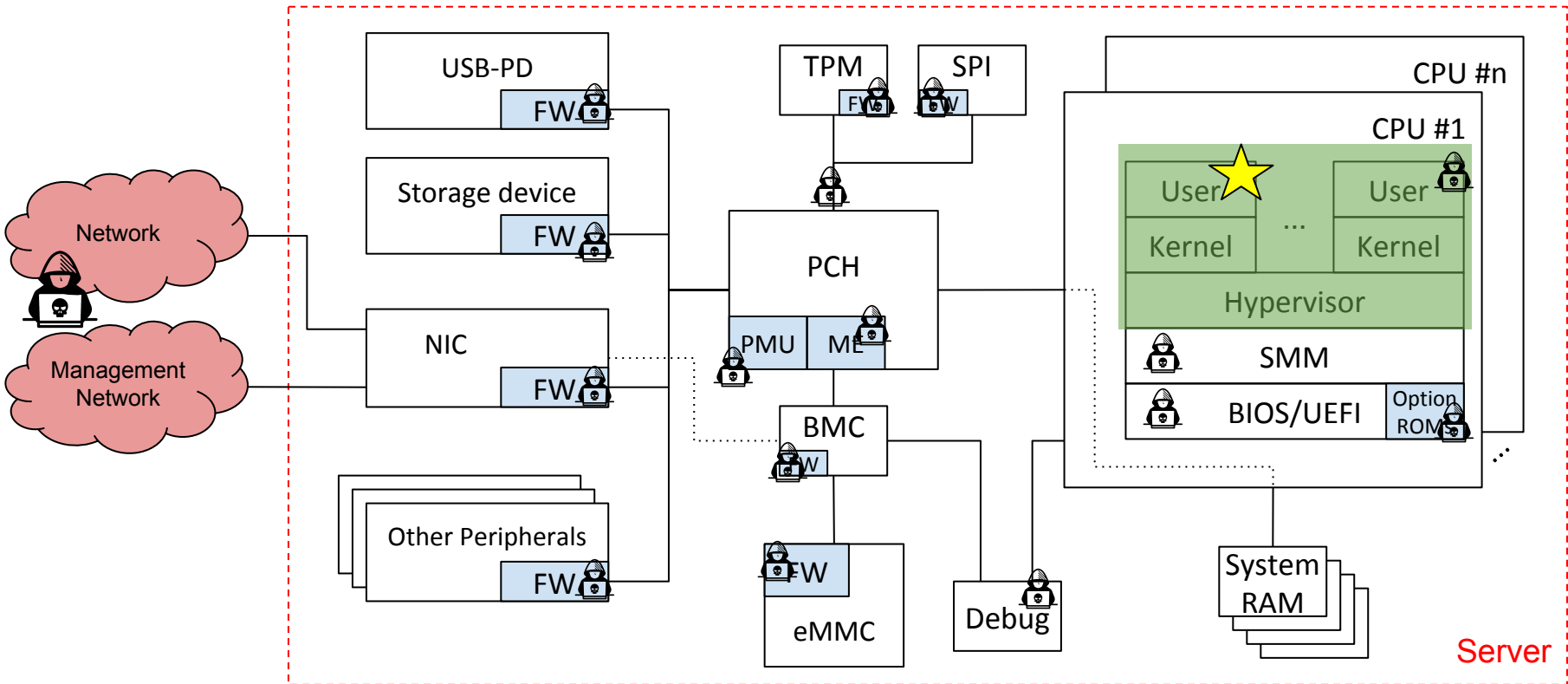
# Server Architecture



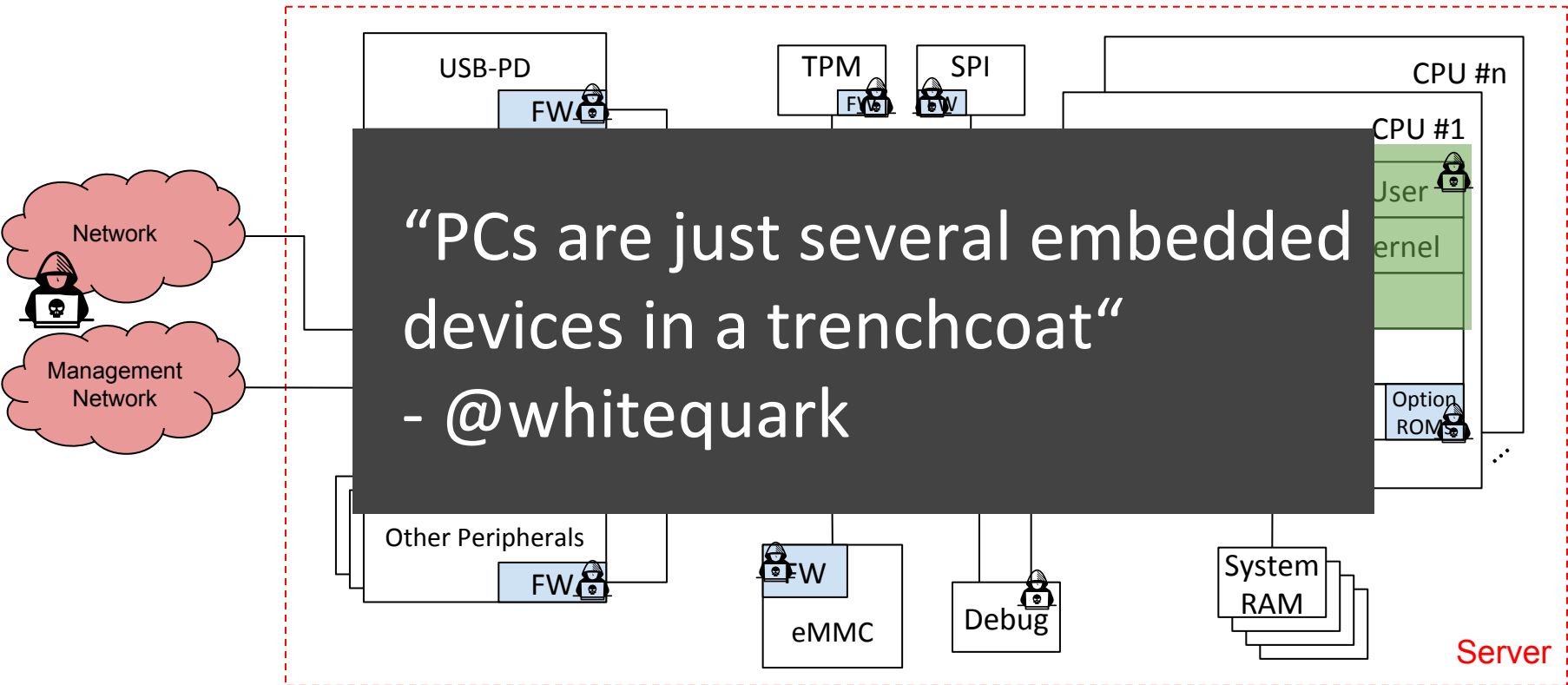
# Server Architecture



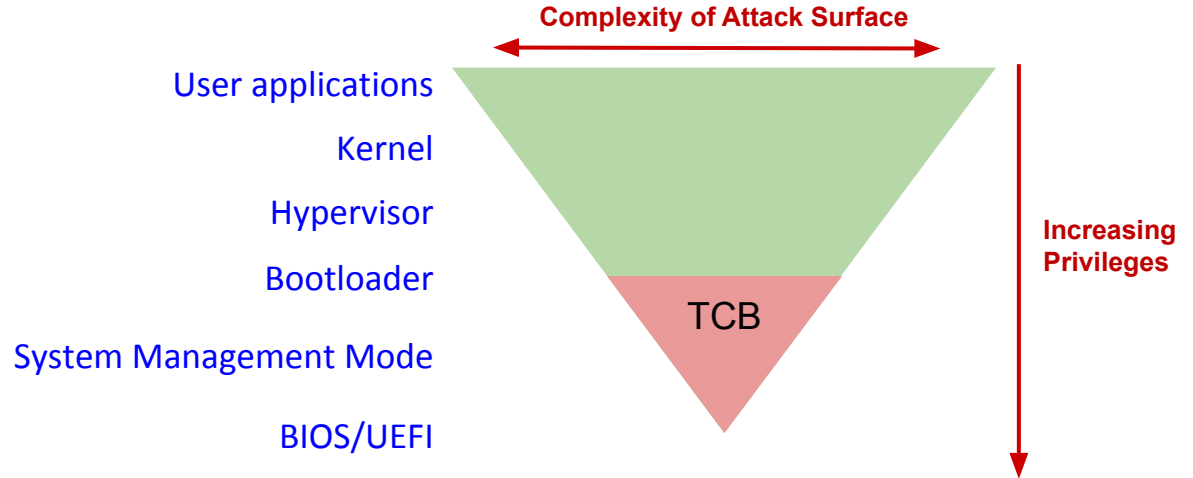
# Server Architecture



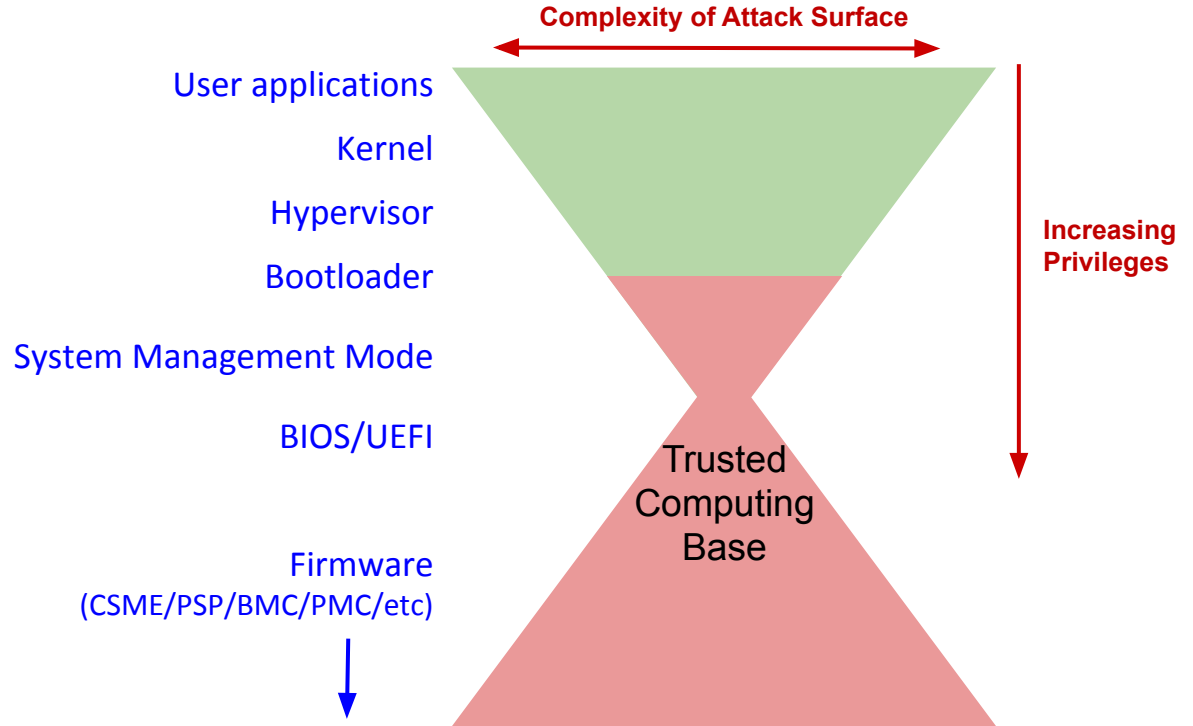
# Server Architecture



# Attack Surface



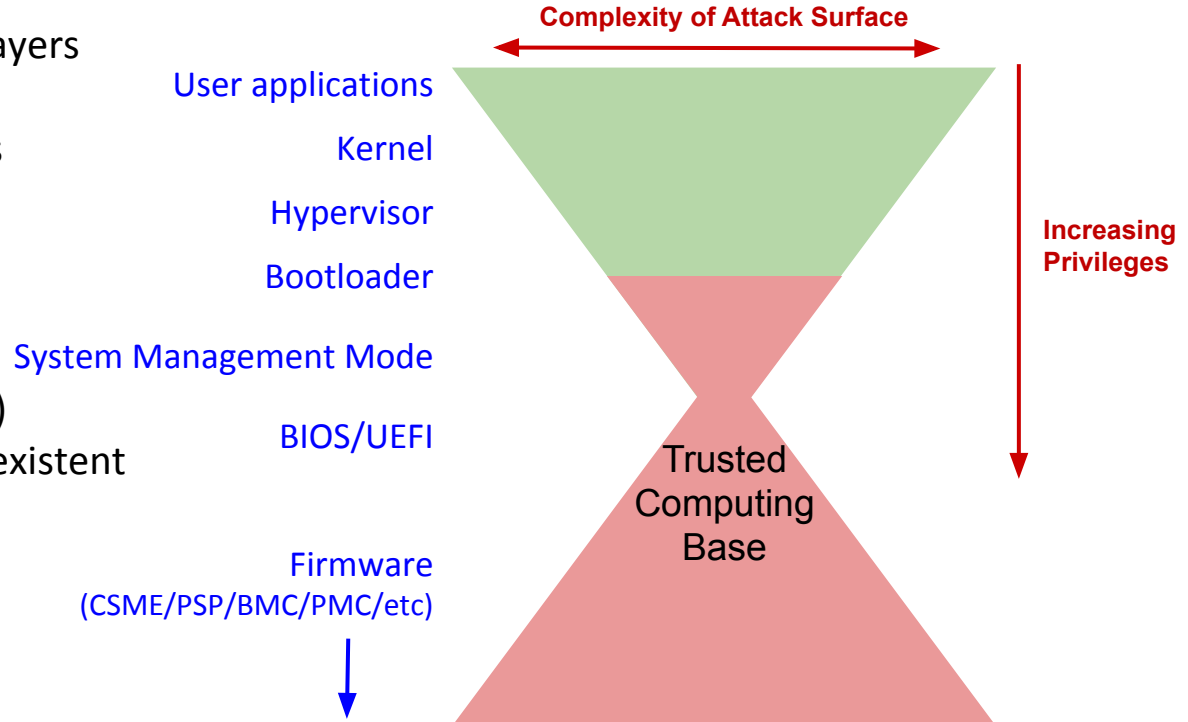
# Attack Surface





# Attack Surface

- Decades of software security improvements, only at upper layers
- Lowest layers written by HW developers, not SW developers
- Complex software stacks
  - often multiple full Linux operating systems!
- Difficulty patching
- 10 year lead time (already old!)
- Documentation is almost non-existent



# Platform Ecosystem Complexity

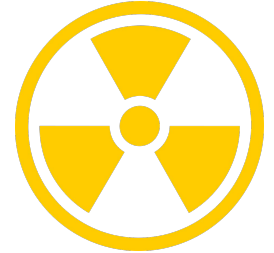
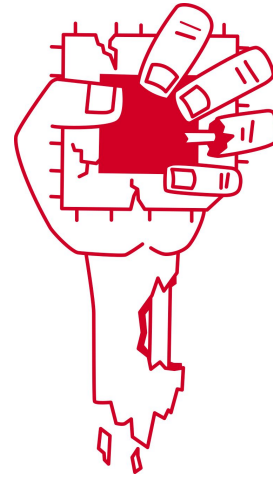
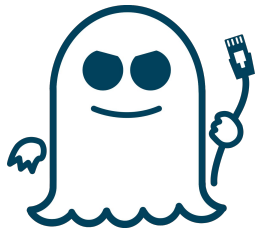
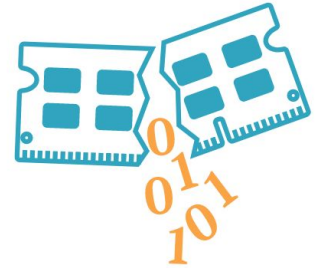
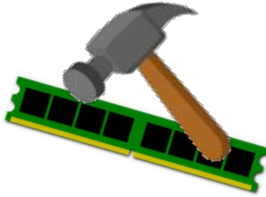
---

- Chip vendors and BSP
- Board OEMs
- Outsource manufacturing
  - Normal stuff, plus “Bloomberg accusations”
  - Interdiction attacks (tamper proof shipping)
- Independent BIOS Vendors (IBVs)
- TPM and PKI providers
  - TPM’s EK is provisioned by TPM vendors
- Hypervisor and OS providers

# Common Platform Issues

# Silicon Issues

- RAM physical side channels
- CPU microarchitectural issues



# Importance of Errata

## Listing of Cortex-A9 revisions

- r0p0
- r0p1
- r1p0
- r1p1
- r1p2
- r2p0
- r2p1
- r2p2
- r2p3
- r3p0

## Listing of Cortex-A15 revisions

- r2p0
- r2p1
- r2p2
- r2p3
- r2p4
- r3p0
- r3p1
- r3p2

## Cortex-A53

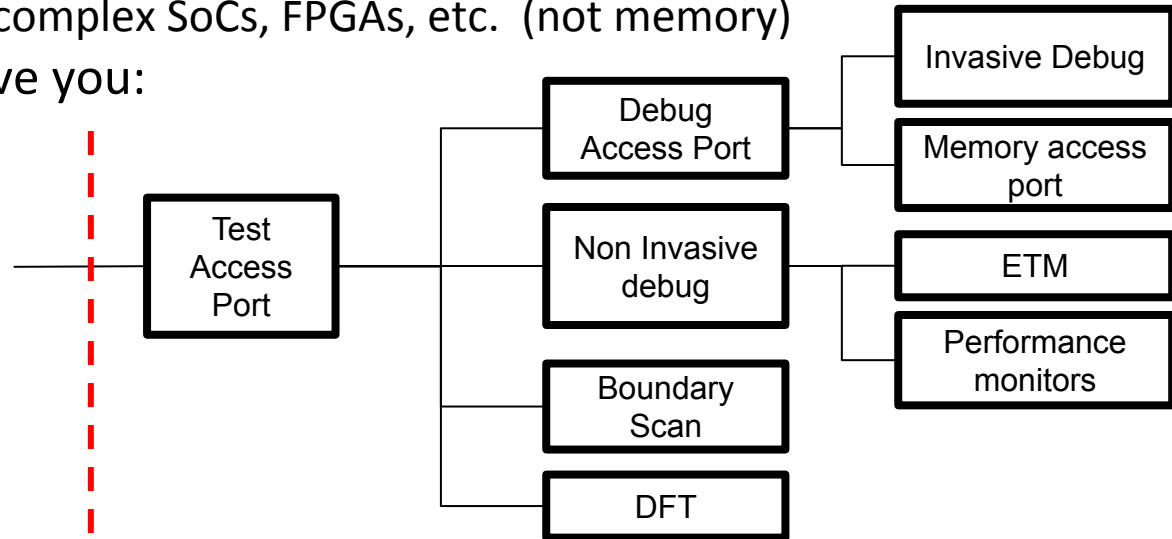
- r0p0
- r0p1
- r0p2
- r0p3
- r0p4

### **571622 - Ability to corrupt SPSR, and return from interrupt into the wrong state**

- Impacts r0p0, fixed in r0p1
- Ability to gain higher privileges by generating error condition and using return from interrupt into higher state than started from
- If exploited, would potentially gain privileged access, and control of the core

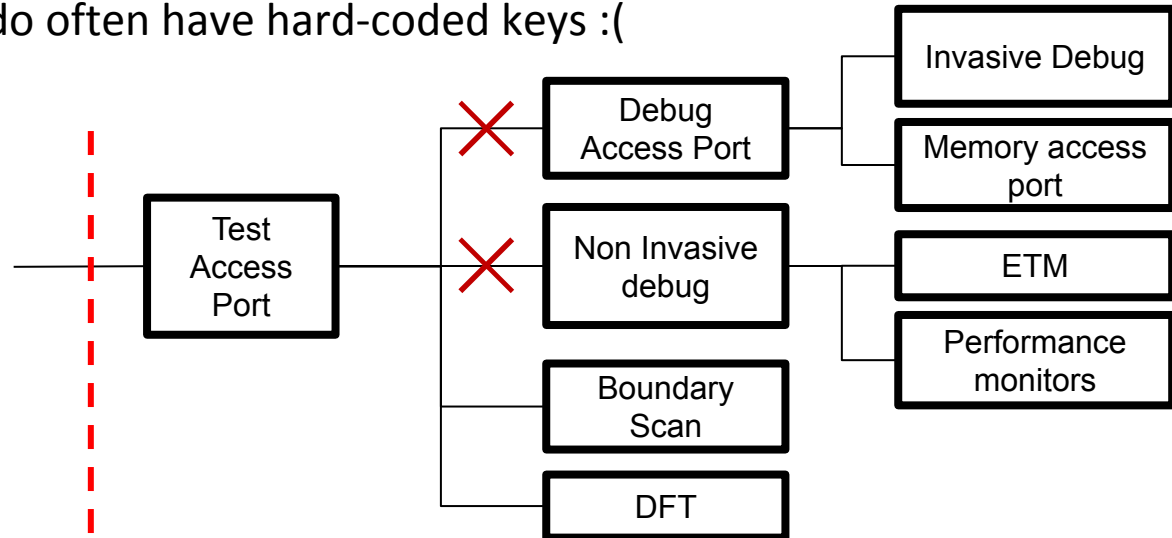
# JTAG

- What is JTAG
  - Manufacturing test
  - Development debug (SWD only does this)
- What devices have it
  - Processors, complex SoCs, FPGAs, etc. (not memory)
- What does it give you:



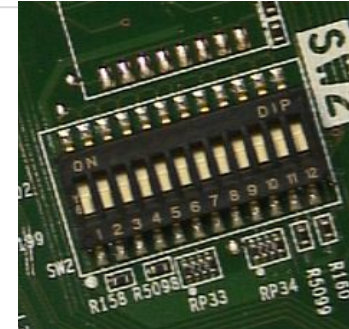
# JTAG - Mitigations

- Fuses (great, but usually not sufficient)
- Physical removal
- Authentication:
  - Some processors can support this :)
  - Those that do often have hard-coded keys :(



# Circuit Issues

- Physical overrides switches/jumpers →
- Unauthenticated debug (JTAG, UART)
- Ineffective anti-tamper mechanisms
- Unprotected bus traffic ↩



## System maintenance switch

Position	Default	Function
S1	Off	Off = iLO security is enabled. On = iLO security is disabled.
S2	Off	Off = System configuration can be changed. On = System configuration is locked.
S3	Off	Reserved
S4	Off	Reserved
S5	Off	Off = Power-on password is enabled. On = Power-on password is disabled.



# Firmware Issues

---

- Configuration issues
  - Mostly a deployment issue, but some OEM issues
    - SoC integration and provisioning
    - RoT key and firmware signing
  - Unprotected configuration is the most common issues we see
  - IOMMU still not configured for most systems
  - Chipsec is awesome! but...
    - Doesn't support AMD or ARM (yet?)
    - Not everyone actually uses it (or it wouldn't find me as many issues)

# Firmware Issues

---

- Configuration issues
- Programmatic bypasses
  - Manufacturing test
  - Recovery features
  - Upgrade functionality

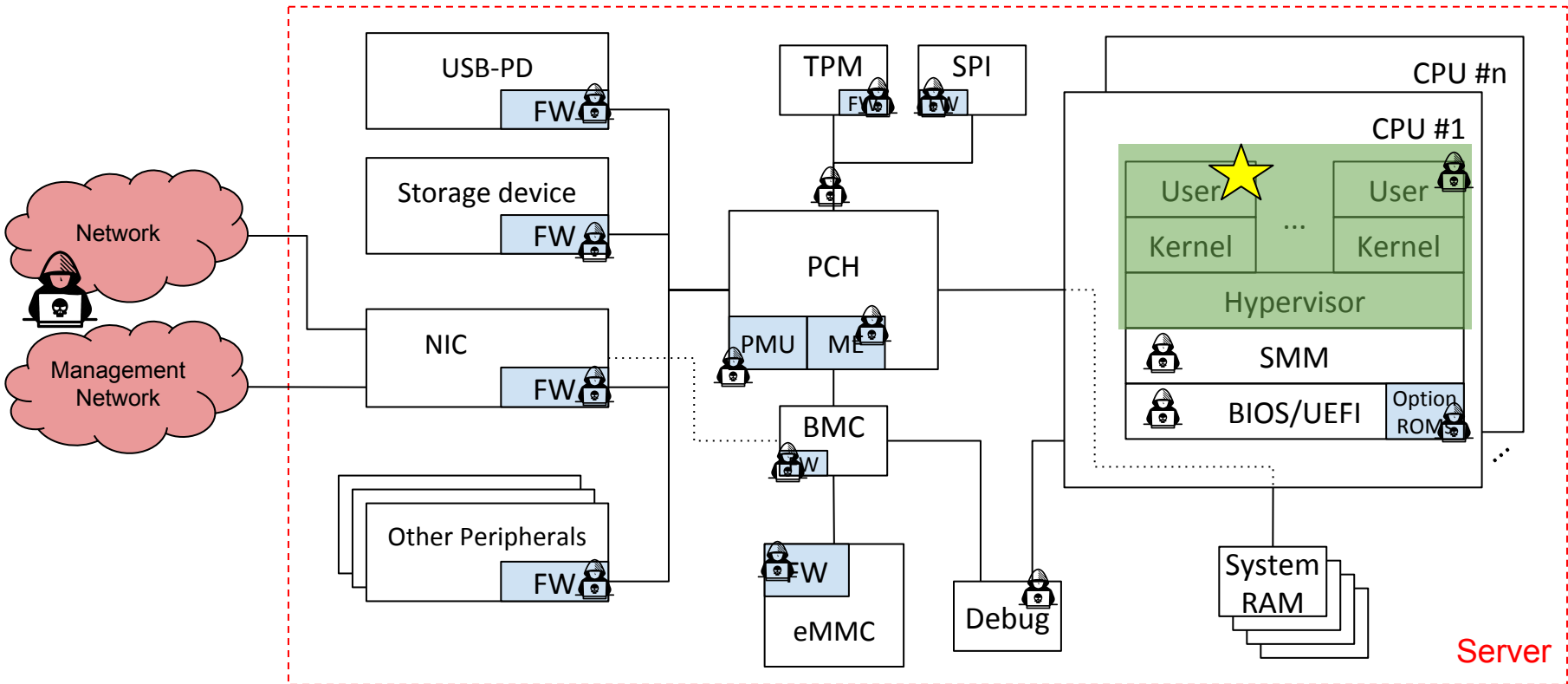
Mostly completely unauthenticated!

Sometimes accessible from the network.

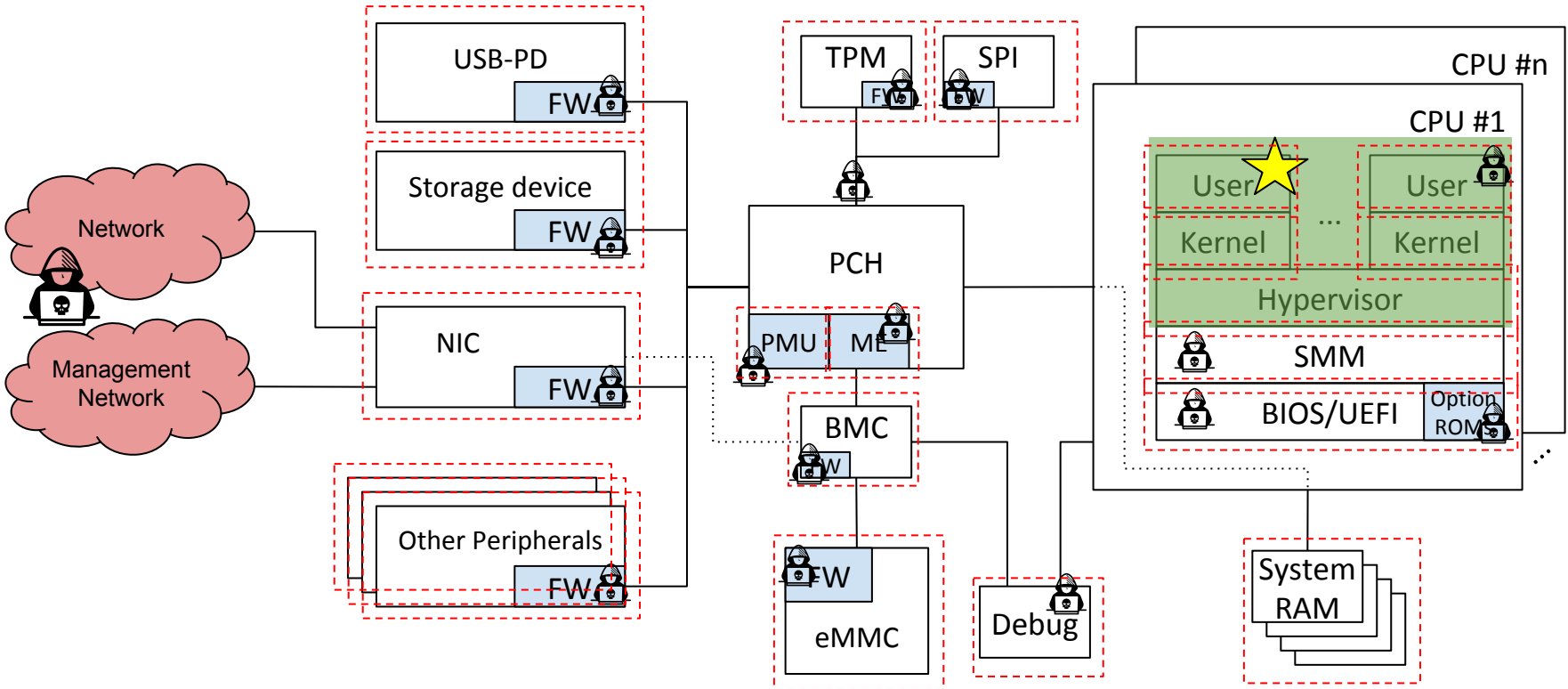
# Firmware Issues

- Configuration issues
- Programmatic bypasses
- Input validation
  - Unchecked lengths
  - Stored addresses/offsets
  - Type confusion (especially between image types, or device families)
  - Not checking the data at the earliest opportunity
  - A general lack of attack surface enumeration (what is an “input”?)
    - Reads from flash memory or DRAM
    - Reads from shared mailbox registers
    - Hardware peripherals
    - Other cores running firmware
    - Other privilege levels on the same core
    - External code written by the same developer

# Server architecture



# Server architecture



# Firmware Issues

---

- Configuration issues
- Programmatic bypasses
- Input validation
- Memory unsafety
  - Reality check: Most firmware is written in C
  - memcpy(dst,src,len)
    - Also DMA copies
    - Also coprocessor/GPU/etc copies
    - Memory aliasing
  - Use-after-free, double-free (but heaps are less common in bare-metal FW)

# Firmware Issues

---

- Configuration issues
- Programmatic bypasses
- Input validation
- Memory unsafety
- Race conditions
  - Multi-threaded firmware
  - Multi-core resource accesses
  - Remember:
    - The faster, lower privileged x86 always wins the race!
    - Probabilistic attacks are valid attacks

# Race Conditions

---

- a.k.a Time-of-Check-Time-of-Use (TOCTOU)
- a.k.a Double-Read or Double-Fetch

Example:

1. Read image from flash to validate data
  2. Read a second time to consume the data
- eXecute-In-Place (XIP) memories
  - Demand-paging applications
  - Poor threat modeling:
    - Attack surface diagram not drawn with enough granularity
    - The flash bus is not secure!



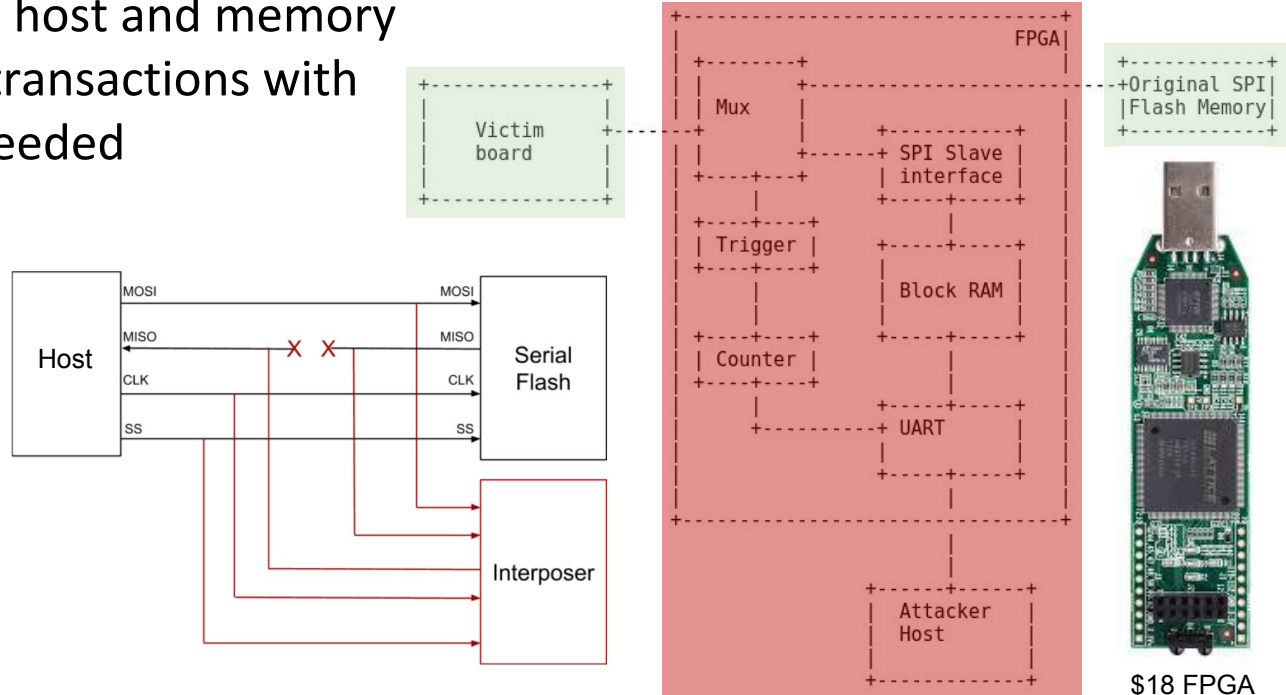
# Race Conditions

## Exploitation:

- Interposer between host and memory
- Selectively replace transactions with malicious ones as needed

## Mitigations:

- Page data into RAM just once!
- Validate data on each read



# Firmware Issues

---

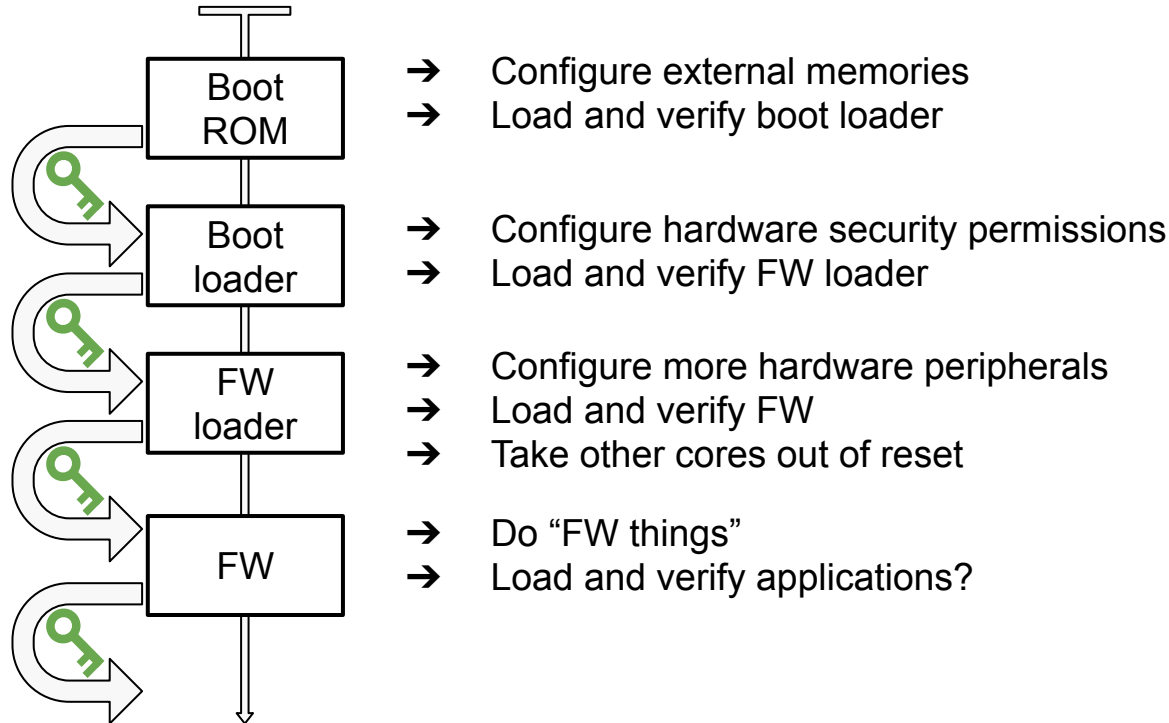
- Configuration issues
- Programmatic bypasses
- Input validation
- Memory unsafety
- Race conditions
- Confused deputy
  - Firmware treats all x86 privilege levels the same
  - Cannot distinguish between legitimate and rogue requests
  - Allows user mode to attack kernel/hypervisor/SMM
  - Example:
    - Coprocessor only distinguishes the core originating a transaction, not the privilege level within that core (PCID/ASID/VMID/etc)

# Firmware Issues

---

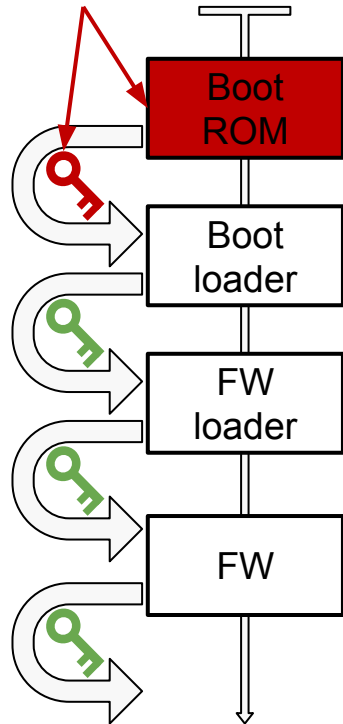
- Configuration issues
- Programmatic bypasses
- Input validation
- Memory unsafety
- Race conditions
- Confused deputy
- **Secure Boot implementation issues**
  - Incomplete or missing chain of trust
  - TOCTOU
  - Partial signing
  - Fail open
  - LinkAddr!=LoadAddr
  - No roll-back prevention

# Secure Boot



# Secure Boot

Immutable\*



- Configure external memories
- Load and verify boot loader
- Configure hardware security permissions
- Load and verify FW loader
- Configure more hardware peripherals
- Load and verify FW
- Take other cores out of reset
- Do “FW things”
- Load and verify applications?

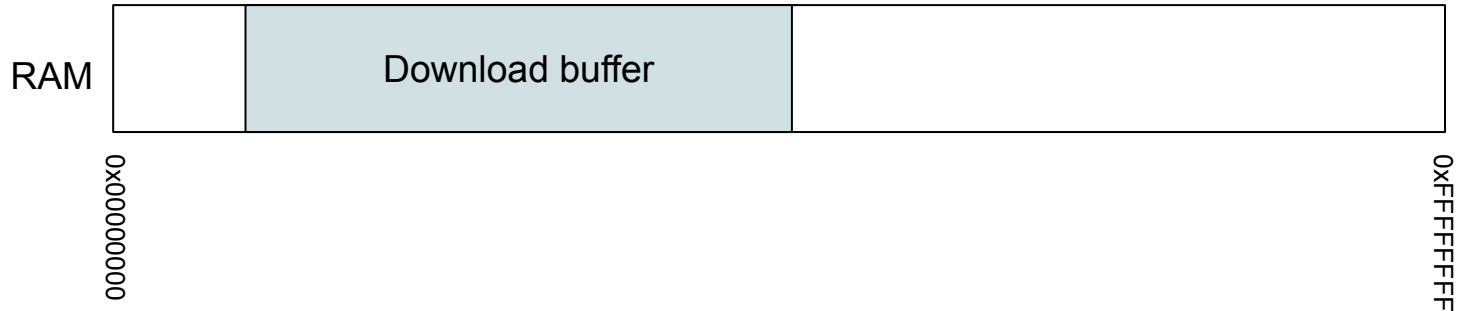
# Trust Anchors

---

- “A public key and the code that uses it”
  - Multiple keys for different purposes/users
- Something internal to the CPU
  - ROM
  - Fuses
  - Internal flash (with careful access controls)
- Not an external component (TPM/SE/EEPROM)
- Not a writable component (eg. Cisco’s FPGA/”Thrangrycat”)

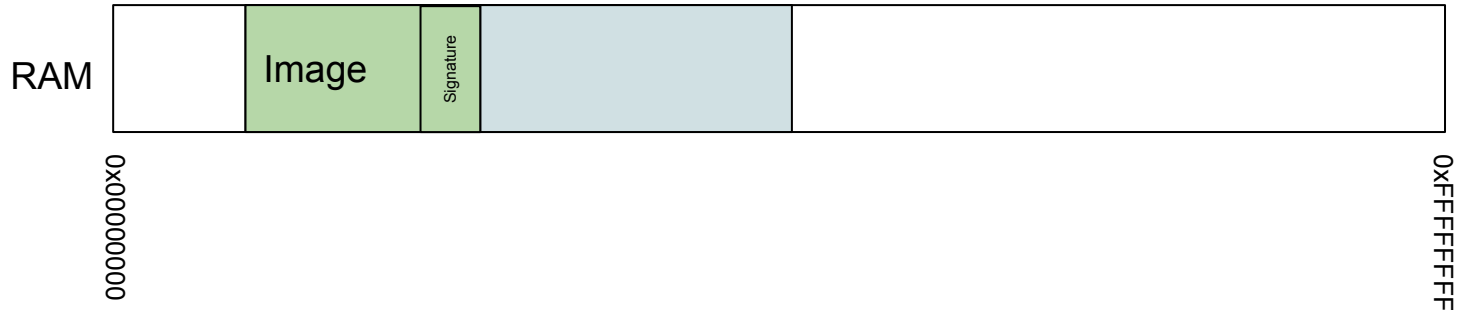
# Secure Boot - LinkAddr

---



# Secure Boot - LinkAddr

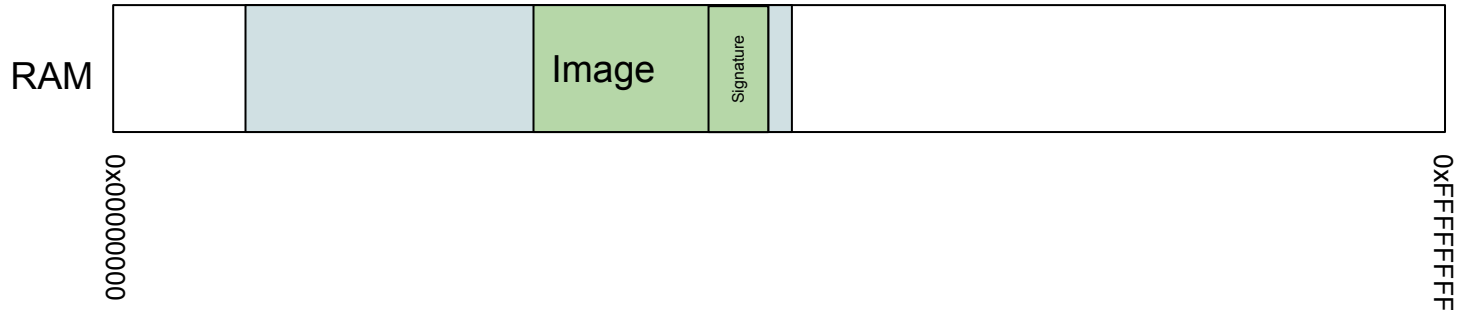
- Signature is valid ✓
- Code is statically linked, with fixed address





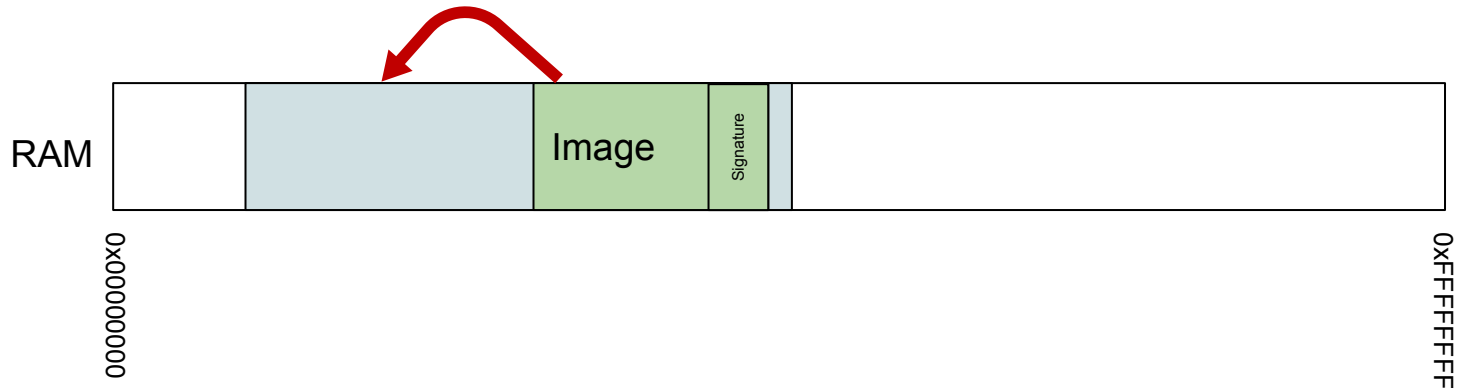
# Secure Boot - LinkAddr

- Signature is valid ✓
- Code is statically linked, with fixed address
- **Attacker controlled LoadAddr**



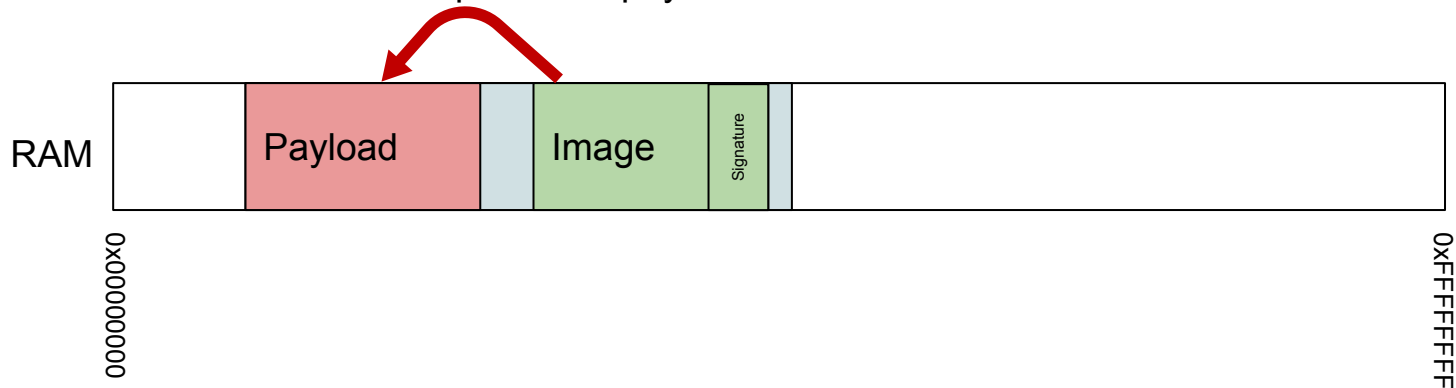
# Secure Boot - LinkAddr

- Signature is valid ✓
- Code is statically linked, with fixed address
- **Attacker controlled LoadAddr**
- Exploitation:
  - Image is loaded at the wrong address
  - First non-relative branch instruction -> code goes off the rails



# Secure Boot - LinkAddr

- Signature is valid ✓
- Code is statically linked, with fixed address
- **Attacker controlled LoadAddr**
- Exploitation:
  - Image is loaded at the wrong address
  - First non-relative branch instruction -> code goes off the rails
    - If attacker can preload a payload to the branch destination



# Supply Chain

# Supply Chain

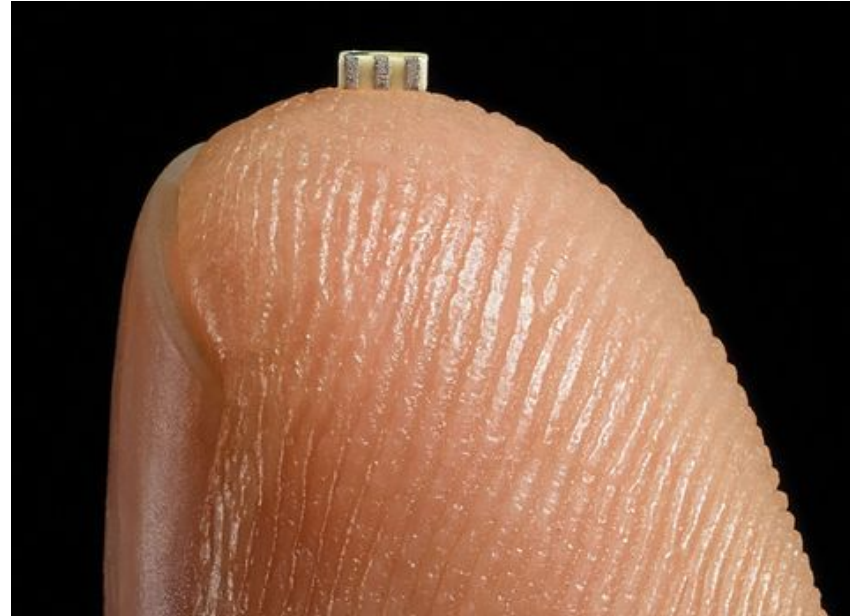
---

As a concept, this is a loaded term:

1. **Global logistics:** Moving “stuff” from one place to another
2. **Factories:** Mostly network and endpoint security
3. **Software supply chain:** 3rd party software dependencies
4. **Product supply chain:**
  - Some overlap with #2 and #3
  - Ensure products can be built in untrusted factories.
  - See [here](#) and [here](#) for some public reading.


# Supply Chain - Examples

1. Bloomberg/Supermicro



# Supply Chain - Examples

1. Bloomberg/Supermicro
2. Google/Feitian BLE security tokens



## FEITIAN

### Product Issue Notification

**Date:** May 15, 2019  
**FEITIAN Technologies**



Dear Valued Customers,

This letter serves as notification that FEITIAN Technologies has become aware of an issue that affects some Bluetooth devices.

**Name of the product**

FEITIAN  
MultiPass FIDO Security Key  
(SKU: K13)

**Nature of the problem**





# Supply Chain - Examples

1. Bloomberg/Supermicro
2. Google/Feitian BLE security tokens
3. Furious Box





# Supply Chain - Detection

- Number of devices: ordered != built != shipped != activated
  - The data needed is not likely in a single system
  - Tracking scrap at each stage can be a problem
  - A few stations vastly overproducing
- Factory network hardening
  - 3rd party factory
  - Station to station traffic
  - TCP packet TTL too high
- Credentials used from wrong site
- Activity during quiet times: local holidays and timezones
- Obsolete devices being newly produced, or produced at the wrong factory



whitepaper

# Supply Chain - Ownership

- Here we specifically mean **control**: “cryptographic ownership”
  - Who signs the firmware?
  - Which firmware? (boot/recovery/debug/etc)
  - Hardware will always need to be trusted at some level (eg. x86 microcode)
- How to provision ownership
  - Early during silicon fabrication/wafer test
  - At OEM during provisioning/onboarding/enrollment/birthing
  - Both is best
- How to transfer ownership
  - Re-sign firmware (signing without audit?!)
  - Does hardware even support changes to the validation keys?
  - Can an attacker install their own keys?

# Final Thoughts

---

- Start with better threat modeling:
  - Understand your FULL attack surface
  - Understand the superset of use cases
- Start with better security requirements:
  - [NIST SP 800-193: Platform Firmware Resiliency](#)
  - [Open Compute Project](#)
- Implement better roots of trust
  - Each chip needs its own!
- Push these costs upstream
  - Embed security requirements and remediation in contracts
  - Revisit current contracts with vendors if possible

# Fin

---

Questions:

[rob.wood@nccgroup.com](mailto:rob.wood@nccgroup.com)

[@finderoffail](#)

